Sentiment Analysis of Social Media Content using N-Gram Graphs

Fotis Aisopos[$], George Papadakis[$,◇], Theodora Varvarigou[$]

[$] ICCS, National Technical University of Athens, Greece {fotais, gpapadis, dora}@mail.ntua.gr
[◇] L3S Research Center, Germany {papadakis}@L3S.de

ABSTRACT

Sentiment Analysis over Social Media facilitates the extraction of useful conclusions about the average public opinion on a variety of topics, but poses serious technical challenges. This is because of the sparse, noisy, multilingual content that is posted on-line by Social Media users. In this paper, we introduce a novel method for capturing textual patterns that inherently supports this challenging type of content. In essence, it creates a graph whose nodes correspond to the character n-grams of a document, while its weighted edges denote the average distance between them. Multiple documents of the same polarity can be aggregated into a polarity class graph, which can be compared with individual documents in order to identify the category of their sentiment. To evaluate our approach, we conducted large scale experiments on a real-world data set stemming from a snapshot of Twitter activity. The outcomes of our evaluation indicate significant improvements over other the methods typically used in this context, not only with respect to effectiveness, but also to efficiency.

Categories and Subject Descriptors

H.3.m [**Information Storage and Retrieval**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Polarity Classification, N-gram Graphs, Social Media

1. INTRODUCTION

Written sentiment expressions denote either the mood of their author or her opinion with respect to a specific entity. The Web is abundant in such expressions, since Web 2.0 technologies enable common users to comment and post online their thoughts about anything. For example, expressing personal feelings is so typical for bloggers that a specialized, emotional search engine, named *We Feel Fine*, has been developed for monitoring them [6].

WSM'11, November 30, 2011, Scottsdale, Arizona, USA.

Most common, though, is the expression of personal feelings through Social Networking Services (SNSs). Networks like Facebook[1] and Twitter[2] have recently acquired a huge share of the overall web activity[3], enabling their users to discuss everyday issues, exchange political views, and evaluate services and products. The extended use of emotional expressions led to the development of specialized notations that signal a feeling, namely the *emoticons* (e.g., ":)" and ":(" for positive and negative feelings, respectively). In a similar vein, the Social Networking platforms developed special functionalities to support the expression of opinions; the "Like" button of Facebook constitutes the most typical example of such functionalities.

In view of the high volume of on-line, user-generated, emotional expressions, numerous serious efforts have been made to leverage on them. A rich diversity of applications can benefit from such approaches; marketing campaigns can receive and evaluate feedback from a large user base, politicians are able to estimate their popularity, manufacturers can identify the drawbacks of their products, while common users are facilitated to navigate the bulk of on-line content. The usefulness of Sentiment Analysis justifies the high number of relevant services that built on Social Media to offer real-time sentiment detection: Twendz[4], Twitter Sentiment[5] and TweetFeel[6], to name but a few.

Existing techniques typically rely on identifying patterns in the free-text that expresses a feeling. These patterns involve either discriminative (series of) words or character n-grams. The former methods are based on the term vector model, while the latter ones on the n-grams model (see Section 3 for more details). Although they achieve high performances in the context of specific settings, they are inadequate for handling the user-generated data of Social Media. The reason is that existing techniques are *language specific*: they are crafted for a certain language, usually relying on a dictionary (like WordNet[7]) for assessing the meaning or the lexical category of specific words or phrases. The textual content of Social Media, however, breaks the fundamental assumption of these methods, due to its inherent characteristics:

1. **Multilinguality**. Although the majority of their users

[1] http://www.facebook.com
[2] http://twitter.com
[3] Alexa, July 2011: http://www.alexa.com.
[4] http://twendz.waggeneredstrom.com
[5] http://twittersentiment.appspot.com
[6] http://www.tweetfeel.com
[7] http://wordnet.princeton.edu

stems from English-speaking countries, SNSs are popular world-wide. Their user base encompasses, therefore, people talking in many different languages and dialects.

2. **Slang and Neologisms**. The textual content of Social Media is rather informal, since it principally comprises communication between fellows. Therefore, users employ words and expressions that are not considered standard in any dialect or language (e.g., "koo" instead of "cool") [2]. In addition, the limited size of their messages (Twitter, for example, allows messages of up to 140 characters) urges them to shorten words into new forms that do not necessarily bear strong similarities to the original one. For example, "gr8" is commonly used instead of "great" and "congratz" instead of "congratulations".

3. **Noise**. The real-time nature of SM encourages users to post their messages quickly, without verifying their correctness with respect its meaning as well as the grammar and the syntactic rules. In case a message (or part of it) is not comprehensible, the author replaces it with a new one. This is the case, for instance, with on-line chats. As a result, the user-generated content abounds in misspelled words and incorrect use of phrases.

All these practices inevitably turn the task of pattern analysis even more complex, calling for a language-neutral method that is tolerant to noise.

In this paper, we propose the use of a new document representation model for the task of content-based sentiment analysis, namely the n-gram graphs. It improves the character n-grams model by adding contextual information: instead of generating a plain bag of n-grams, it considers the sequence of appearance to model a document as a graph. Its nodes correspond to specific n-grams, and the edges connecting them signal how close they are found - on average - in the given document. They, thus, capture more information than n-grams, while making no assumptions on the language of the documents at hand. This leads to higher effectiveness, which is combined with higher efficiency, as well: the number of features they employ does not rely on the diversity of the vocabulary of the given documents. Instead, its features depend solely on the number of polarity classes, with each class introducing three different similarity measures. Thus, unlike the other representation models, the n-gram graphs one does not suffer from the dimensionality curse.

In summary, the main contributions of the paper can be summarized as follows:

- We explain how n-gram graphs model can be used to effectively capture patterns in the polarized, textual content of Social Media.
- We explain the advantages of the n-gram graphs over the established document representation models, both with respect to effectiveness and efficiency.
- We conduct a comparative analysis between the three representation models, evaluating their performance over a large-scale collection of real-world data from Twitter. The outcomes of our study verify the higher performance conveyed by the n-gram graphs.

The rest of the paper is structured as follows: Section 2 formally defines the problem we are tackling, and Section 3 introduces our approach, comparing it with the established methods of content-based sentiment analysis. Section 4 presents our thorough experimental evaluation on a large-scale, real-world data set, and Section 5 discusses and categorizes appropriately previous work in the field. Finally, Section 6 concludes the paper along with directions for future work.

2. PRELIMINARIES

In the following, we focus on the micro-blogging service of Twitter as the test-bed for evaluating our approach. There are several reasons for this choice: first, it entails strict rules for social interaction, as it encompasses a limited - yet extremely flexible and expressive - way of communicating opinions and emotions; users are only allowed to post short messages of free text of up to 140 characters, which are called *tweets*. In contrast, other Social Media offer a more diverse set of interactions between users, thus complicating the study of content-based sentiment analysis. Second, Twitter offers easy ways to access great volumes of real-world, user-generated data through its handy API. There is, also, a standard and well-established representation model (i.e., emoticons) for data, which allows for efficiently extracting the features of interest from debates on a topic. Last but not least, Twitter is one of the most popular Social Media, comprising a user base of around 200 million users that post 1 billion short messages per week[8]. To its success also attest the large number of specialized services that have been developed on top of it, like twitrratr[9], an application that tracks opinions on Twitter. These characteristics explain why Twitter currently lies on the focus of intensive research.

Our analysis mainly considers the *polarized tweets*; that is, the tweets that express either a positive or a negative sentiment, as denoted by the appropriate emoticon. In more detail, we consider as a **positive tweet** one that contains either of the following smileys: ":)", "(:", ":-)", "(-:", ":)", "(:", ":D" or "=)". On the other hand, we classify as **negative tweets** those that are a marked with ":(", "):", ":-(", ")-:", ": (" or ") :". Tweets that lack any polarity indicator are considered to express neither a positive nor a negative sentiment (i.e., **neutral** tweets). Note that in our analysis we completely disregard tweets that contain both a positive and a negative emoticon (i.e., they are considered neither polarized nor neutral). These assumptions are common practice in the related literature [5, 7, 10].

2.1 Problem Formulation

Sentiment analysis is typically modeled in the literature is as binary classification problem. In fact, it is treated as a *single-labeled classification* problem: each document (i.e., tweet) belongs to a single polarity category. Thus, the goal is usually to identify whether a tweet is negative or positive. Based on the above definitions, this problem can be formally defined in the context of Twitter as follows:

Problem 1 (Binary Polarity Classification). *Given a collection of tweets $\mathcal{T}$ and the set of binary polarity classes $\mathcal{P}_\mathcal{B} = \{negative, positive\}$, the goal is to approximate the unknown target function $\Phi : \mathcal{T} \rightarrow \mathcal{P}_\mathcal{B}$, which describes the polarity of tweets according to a golden standard,*

[8]See `http://blog.twitter.com/2011/03/numbers.html` for more details.

[9]`http://twitrratr.com`.

by means of a function $\Phi' : \mathcal{T} \rightarrow \mathcal{P}_{\mathcal{B}}$*, which is called the* ***binary polarity classifier****.*

This settings simplify the problem of sentiment analysis, resulting in higher classification accuracy. However, the input of real-world sentiment analysis applications rarely consists of just polarized tweets. Practical methods have to consider, therefore, the additional class of neutral tweets. The following, more general problem of polarity classification has to be taken into account, as well:

Problem 2 (General Polarity Classification). *Given a collection of tweets* $\mathcal{T}$ *and the set of general polarity classes* $\mathcal{P}_{\mathcal{G}} = \{negative, neutral, positive\}$*, the goal is to approximate the unknown target function* $\Phi : \mathcal{T} \rightarrow \mathcal{P}_{\mathcal{G}}$*, which describes the polarity of tweets according to a golden standard, by means of a function* $\Phi' : \mathcal{T} \rightarrow \mathcal{P}_{\mathcal{G}}$*, which is called the* ***general polarity classifier****.*

Existing works addressing Problem 2 invariably split it in two steps: the first one aims at categorizing documents into subjective and objective (i.e., neutral) ones, while the second one further distinguishes subjective documents into positive and negative ones. We decided to address both of these sub-problems simultaneously in the context of Problem 2 for two reasons: first, we provide in this way a holistic overview of the performance of Sentiment Analysis and, second, we illustrate the effect of considering an additional class in Problem 1.

3. DOCUMENT REPRESENTATION MODELS FOR SENTIMENT ANALYSIS

In this section, we first present the document representation models that are typically employed in the context of content-based sentiment analysis. We analyze their differences and explain their weaknesses, when applied to the multilingual, noisy content of Twitter and the other Social Media. We then introduce a novel technique for capturing textual patterns, and elaborate on its technical characteristics that turn it suitable for polarity classification.

3.1 Term Vector Model

This method constitutes the cornerstone of Information Retrieval as the dominant technique for detecting the documents that are most relevant to a keyword query [8]. In the context of text classification and, thus, sentiment analysis, it is employed as follows: given a collection of tweets $\mathcal{T}$, it aggregates the set of distinct words (i.e., tokens) $\mathcal{W}$ that are contained in it. Each tweet $t_i \in \mathcal{T}$ is then represented as a vector $\mathbf{v_{t_i}} = (v_1, v_2, \ldots, v_{|\mathcal{W}|})$ of size $|\mathcal{W}|$ with its j-th dimension v_j quantifying the "importance" of the j-th token $w_j \in \mathcal{W}$ for t_i. This measure is actually expressed through TF-IDF weights: the value of each term w_i is defined as the product of its *Term Frequency* TF_i and its *Inverse Document Frequency* (IDF_i). The former denotes the number of times the corresponding term occurs in a specific document (i.e., tweet). The latter encapsulates the cross-document frequency of terms in order to degrade the weight of tokens that appear in many tweets (e.g., stop words). In more detail, it is defined as follows: $DF_i = \log \frac{|\mathcal{T}|}{|\{t : w_i \in t \wedge t \in \mathcal{T}\}|}$, where the numerator stands for the size of the input collection, and the denominator expresses the number of tweets that contain the word w_i [8].

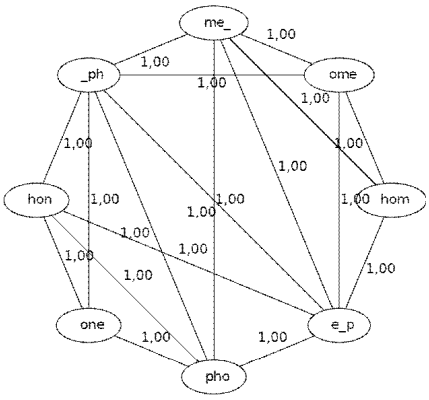

Figure 1: Sample tri-gram graph representing the string "home_phone". Every tri-gram of this string corresponds to a node in the graph, and the edges connect tri-grams, whose distance is less than three letters, regardless of their relative position.

Similar to individual documents (i.e., tweets), polarity classes are modeled as term vectors, which comprise the tokens that have been aggregated from the tweets they entail.

3.2 Character N-Grams

The set of *character n-grams* of a document comprises the substrings of length n of the original text. The most common sizes for n are 2 (*bigrams*), 3 (*trigrams*) and 4 (*four-grams*). For example, the phrase "home_phone" consists of the following tri-grams: hom, ome, me_, _ph, pho, hon, one. According to this model, each tweet is represented by a vector whose $i-th$ dimension encapsulates the importance of the $i-th$ n-gram. Instead of TF-IDF weights, the n-grams are typically used in conjunction with the term frequency of the corresponding n-grams. Similarly, a polarity class is modeled as a vector that comprises all the n-grams contained in its tweets.

The main advantage of this model over the previous one is its tolerance to noise and spelling mistakes: by considering substrings instead of entire words, the likelihood of serious spelling mistakes is significantly reduced.

3.3 N-Gram Graphs

The main drawback of the previous model is that it converts a tweet into a bag of n-grams, thus disregarding the valuable information that is encapsulated by the actual position of the n-grams in the original text. For instance, the words "wiki" and "kiwi" have the same bigrams representation, although their meaning is totally different.

To overcome this problem, the *n-gram graphs* method, which was initially coined in [3] as a summarization method, associates all pairs of n-grams with edges that denote how close they are found on average in the given tweet(s). That is, it forms a graph whose nodes correspond to distinct n-grams, while its edges are weighted proportionally to the average distance - in terms of n-grams - between the respective nodes. To illustrate this structure, Figure 1 depicts the n-gram graph derived from the phrase "home_phone". Apparently, it conveys more information than the trigram representation of the same phrase in Section 3.2.

More formally, an n-gram graph is defined as follows [3]:

Definition 1 (N-Gram graph). *An n-gram graph is a graph* $G = \{V^G, E^G, W\}$*, where* V^G *is the set of vertices (labeled by the corresponding n-gram),* E^G *is the set of di-*

rected edges (labeled by the concatenation of the labels of its vertices in the direction of the connection), and W is a function assigning a weight to every edge.

Note that an n-gram graph is characterized by three parameters: (i) the minimum n-gram rank $L_{\min}$, (ii) the maximum n-gram rank L_{MAX}, and (ii) the maximum neighborhood distance D_{win} [3]. In the following, we exclusively consider the configuration of $L_{\min} = L_{\text{MAX}} = D_{\text{win}} = n$, which was experimentally verified to provide a performance close to the optimal one, as derived after detailed fine-tuning [3].

To construct an n-gram graph, a window of size D_{win} is run over a given tweet, analyzing it into overlapping character n-grams and recording information about the neighboring ones (within the window). Thus, an edge $e^G \in E^G$ connecting a pair of n-grams indicates proximity of these character sequences in the original text within the predefined window of size D_{win} [3]. The actual weight of the edges is estimated by measuring the percentage of co-occurrences of the vertices n-grams within the window.

The n-gram graphs model can be used to uniformly represent a set of tweets (i.e., polarity class) through a single graph. This is simply done with the help of the *update functionality* [4]: given a set of tweets, $\mathcal{T}$, it builds an initially empty graph G_T. The $i-th$ tweet $t_i \in \mathcal{T}$ is then transformed into an n-gram graph G_{t_i} that is merged with G_T to form a new graph G^u with the following properties: $G^u = (E^u, V^u, W^{u_i})$, where $E^u = E^{G_T} \cup E^{G_{t_i}}$, $V^u = V^{G_T} \cup V^{G_{t_i}}$ and $W^{u_i}(e) = W^{G_T}(e) + (W^{G_{t_i}}(e) - W^{G_T}(e)) \times 1/i$ (as explained in [4], the division by i in the computation of the new weights ensures that the aggregated weight converges to the mean value of the individual ones).

With the help of the update functionality, we can combine all tweets (of the training set) of each polarity class into a common graph (i.e., one class for the negative tweets, one for the neutral and one for the positive ones). The resulting graphs capture patterns common in the content of each class, such as recurring and neighboring character sequences, special characters, and digits. Thus, they can be employed to measure the similarity of a single tweet (of the testing set) with each polarity class. The similarity is calculated between the corresponding graph representation G^i of the tweet and a class representative graph G^j. The following three kinds of similarity between n-gram graphs are used in the scope of this work:

(i) **Containment Similarity (CS)** expresses the proportion of edges of a graph G^i that are shared with a second graph G^j. Assuming that G is an n-gram graph, e is an n-gram graph edge and that for function $\mu(e, G)$ it stands that $\mu(e, G) = 1$, if and only if $e \in G$, and 0 otherwise, then: $\text{CS}(G^i, G^j) = \frac{\sum_{e \in G^i} \mu(e, G^j)}{\min(|G^i|, |G^j|)}$, where $|G|$ denotes the number of edges of graph G (i.e., the *size of the n-gram graph*).

(ii) **Size Similarity (SS)** denotes the ratio of sizes of two graphs: $\text{SS}(G^i, G^j) = \frac{\min(|G^i|, |G^j|)}{\max(|G^i|, |G^j|)}$.

(iii) **Value Similarity (VS)** indicates how many of the edges contained in graph G^i are contained in graph G^j, considering also the weights of the matching edges. In this measure, each matching edge e having a weight $W^i(e)$ in graph G^i contributes $\frac{\text{VR}(e)}{\max(|G^i|, |G^j|)}$ to the sum, while not matching edges do not contribute (consider that for an edge $e \notin G^i$ we define $w_e^i = 0$). The *ValueRatio (VR)* is a scaling factor that is defined as: $\text{VR}(e) = \min(w_e^i, w_e^j)/\max(w_e^i, w_e^j)$. The equation indicates that the ValueRatio is symmetric, taking values in the interval $[0, 1]$. Thus, the full equation for *VS* is: $\text{VS}(G^i, G^j) = \frac{\sum_{e \in G^i} \frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)}}{\max(|G^i|, |G^j|)}$. This measure converges to 1 for graphs that share both the edges and similar weights, with a value of VS = 1 indicating perfect match between the compared graphs.

An important, derived measure is the **Normalized Value Similarity (NVS)**, which is computed as: $\text{NVS}(G^i, G^j) = \frac{\text{VS}(G^i, G^j)}{\text{SS}(G^i, G^j)}$. The NVS is a measure of value similarity where the ratio of sizes of the two compared graphs does not play a role.

On the whole, to classify a tweet according to the n-gram graphs model, we simply need to consider three similarities for each one of the involved classes. That is, the CS, the VS, and the NVS for the positive and the negative class in the Binary Polarity Problem and the same similarities for the neutral class in the General Polarity Problem.

Discretizing N-Gram Graph Similarities.

The above similarities can take very low values; for example, the containment similarities of a particular tweet with the negative and the positive might differ only in 8th decimal point. For some classifiers, this is too fine a difference to identify reliable patterns. In this cases, a discreet value can be used instead to provide more clear evidence for the correct class of a tweet. Thus, we can discretize the similarities of a single tweet with respect to two polarity classes of Problem 1 as follows:

$$dsim(sim_{neg}, sim_{pos}) = \begin{cases} negative, & \text{if } sim_{pos} < sim_{neg} \\ equal, & \text{if } sim_{neg} = sim_{pos} \\ positive, & \text{if } sim_{neg} < sim_{pos} \end{cases}$$

Note that this discretization technique is applied on similarities of the same kind (e.g., by comparing the CS of the negative class with the CS of the positive one). As a result, a tweet is classified in Binary Polarity problem according to three nominal features: $dsim(CS_{neg}, CS_{pos})$, $dsim(NVS_{neg}, NVS_{pos})$, and $dsim(VS_{neg}, VS_{pos})$.

In the case of Problem 2, six more nominal features are added, by comparing the similarities of the neutral class with the respective ones of the negative - $dsim(CS_{neg}, CS_{neu})$, $dsim(NVS_{neg}, NVS_{neu})$, and $dsim(VS_{neg}, VS_{neu})$ - and the positive class - $dsim(CS_{neu}, CS_{pos})$, $dsim(NVS_{neu}, NVS_{pos})$, and $dsim(VS_{neu}, VS_{pos})$.

4. EVALUATION

Data set. In our experimental study, we employed a real-world data set of more than 475 million tweets that were posted by over 17 million users in a time period of 7 months (from June 2009 until December 2009) [15]. It contains 6.12 million negative and 14.12 million positive tweets, as they are defined by the emoticons mentioned in Section 2. To create equally sized samples, we randomly selected 1 million tweets from each polarity category, along with 1 million neutral tweets for Problem 2. Note that this is one of the largest data sets employed so far in the context of sentiment analysis over Twitter.

Evaluation Setup. Our evaluation of the n-gram graphs model relied on a 10-fold cross validation scheme: in each

		Vector Model	2-Grams	3-Grams	4-Grams	2-Gram Graphs		3-Gram Graphs		4-Gram Graphs	
						Sim.	Discr.	Sim.	Discr.	Sim.	Discr.
(a)	**NBM**	62.44%	59.44%	62.94%	64.76%	57.15%	-	61.36%	-	64.39%	-
	C4.5	58.37%	56.85%	59.24%	60.98%	60.06%	58.50%	63.07%	61.87%	66.77%	65.34%
(b)	**NBM**	44.97%	44.89%	46.97%	47.41%	40.97%	-	43.94%	-	46.14%	-
	C4.5	41.68%	40.74%	43.53%	44.23%	44.85%	45.95%	47.33%	48.69%	50.67%%	51.31%
(c)	**Pr. 1**	148	589	1,456	1,041	6	3	6	3	6	3
	Pr. 2	142	596	1,448	1,011	9	9	9	9	9	9

Table 1: (a) Success rate of all models on the Binary Polarity Classification problem, (b) Success rate of all models on the General Polarity Classification problem, and (c) Number of features employed by each method for each problem.

of the 10 iterations, 90% of the tweets of each polarity class composed the training test, and the remaining 10% formed the testing set. For each class, half of the training set was combined into an aggregate graph, which was then compared with the entire data set. The resulting similarity values comprised the input features of the classification algorithms.

For the vector and the n-gram models, we employed a similar 10-fold cross validation scheme; at each rotation, 90% of the data set comprised the training set, with half of it used for deriving the classification features. More specifically, we estimated the frequency of the distinct tokens and substrings (respectively) inside the training set and, similar to [1], we filtered out those items that do not appear in at least 1% of the training elements.

Metrics. To estimate the effectiveness of the above models, we employed the metric of *accuracy*; in the settings we are considering (i.e., single-label classification problems), it is simply defined as the ratio of correctly classified documents over the size of the entire document collection.

Effectiveness Performance Analysis. The goal of this analysis is to twofold: first, to compare the n-gram graphs with the vector model and the n-grams one, and, second, to identify their best performing configuration (i.e., the optimal size for n).

We employed two classification algorithms: the *Naive Bayes Multinomial* (**NBM**), which is a method typically used in text classification, and the C4.5 tree classifier, which is more appropriate for the similarity features of the n-gram graphs[10]. For a detailed description of these algorithms see [14]. For their implementation, we relied on Weka[11]. The implementation of the n-gram graphs was provided by the open-source library JInsect[12]. Note that we employed the default configuration of C4.5, without any fine tuning. It should also be stressed that the NBM is not applicable to instances described only by nominal attributes, such as the ones resulting from the discretization of n-gram graphs similarities. Finally, it is worth clarifying that the vector model did not involve any fine-tuning processing, like stemming or lemmatization; such approaches are not practical due to the rich diversity of languages contained in our corpus.

The outcomes of our experimental study are presented in Tables 1(a) and (b). We can easily notice that the n-grams outperform the vector model for $n = 3$ and $n = 4$ in all the cases. Bigrams, on the other hand, have an accuracy that is slightly lower than that of the vector model. This behavior is a probably caused by the advantages of n-grams over the vector model: they are language-neutral and robust to noise. These advantages, however, are not fully exploited in the case of bigrams; their low performance indicates that their size is not sufficient for capturing reliable patterns in multilingual settings.

Regarding the n-gram graphs, we can see that, unlike the other models, they achieve relatively low accuracies for the NBM, but take substantially higher values for the C4.5. In the latter case, actually, they achieve the highest performance among all content-based models, with four-gram graphs being the overall champion. Three patterns are worth noting: first, the performance of the n-gram graphs increases by 3%-4% when the size of n is incremented by one. Second, their performance is consistently higher than that of the corresponding n-grams model, probably because of the additional, contextual information they capture. Although the differences are not very high ($\leq 6\%$), they were found to be statistically significant ($\rho < 0.005$) in all the cases. Third, the performance of the discretized n-gram graphs similarities is consistently lower than that of the actual similarity values for Problem 1, and vice versa for Problem 2, where they actually achieve the highest overall accuracy. This can be easily explained by the number of features involved in each case: as shown in Table 1(c), for Problem 1, the six numeric similarities are replaced by 3 nominal ones, inevitably losing significant information. For Problem 2, though, the 9 numeric attributes are replaced by an equal number of nominal ones, which adequately capture the relevant patterns.

Efficiency Performance Analysis. For a practical, large-scale application of polarity classification, the number of features plays an important role; the more features are required, the higher is the computational cost, not only for the training phase, but also for the testing one. The reason is that the search space grows exponentially with the increase of its dimensions, even if the instances that are considered are sparse: the number of features is more critical for the classification efficiency than the distribution of their values. Although the required number of features can be limited by filtering algorithms like PCA, approaches that inherently rely on a limited number of features are more suitable for large-scale classification applications.

To study the efficiency of the aforementioned representation models, we present in Table 1(c) the number of features they entail. We can easily notice that the n-grams involve the largest by far set of features, with the peak corresponding to $n = 3$. This is because, as the size of n increases,

[10]Note that the goal of our analysis is to compare the individual models on an equal basis, rather than trying to achieve the highest possible performance. This is why we did not use the state-of-the-art text classification algorithm: SVMs. They require an extensive fine-tuning to optimize their performance, which is not practical at the scale of data we are considering. NBM, on the other hand, is equally popular and requires no fine-tuning.

[11]`http://www.cs.waikato.ac.nz/ml/weka`

[12]`http://sourceforge.net/projects/jinsect`

the number of possible combinations grows exponentially. Nevertheless, the four-grams entail less features than the trigrams, because their numerous substrings are rather rare and a smaller part of them exceeds the frequency threshold. As expected, the vector model involves less features than the n-grams, as it considers entire words instead of substrings. Most notably, though, the n-gram graphs require a significantly lower number of features in all cases (<10), thus achieving a significantly higher classification efficiency.

5. RELATED WORK

Numerous works have investigated ways of exploiting the average sentiment of Twitter in order to predict the popularity of specific politicians or political parties. They typically rely on dictionaries that associate polarized words with certain sentiments and perform opinion mining simply by measuring their frequency. O'Connor et al. [9], for instance, examined the correlation between telephone public opinion polls and the aggregate sentiment in Twitter with respect to two different topics: consumer confidence and political opinions during the 2008 U.S. Presidential Elections. They inferred the polarity of individual tweets from the subjectivity lexicon of OpinionFinder [13] and demonstrated that the resulting, average sentiment score per day provided a close estimation for the evolution of the public poll outcomes.

Similarly, Tumasjan et al. [12] investigated the accuracy of predicting political sentiment through Twitter in the case of the 2009 German elections. They analyzed 100,000 relevant tweets, using the LIWC text analysis methodology [11] to extract the sentiment of individual tweets. At the core of their approach lies a set of predefined psychological and structural categories to which specific words are mapped. The outcomes of their analysis suggests that the mere portion of tweets that mention a political party reflects the actual vote share with remarkable accuracy.

Another line of research focuses on content-based, supervised machine learning techniques that determine a document's sentiment without requiring a dictionary. Lake [7], for instance, considers the term vector model, presenting a series of elaborate feature selection approaches. The resulting feature space is applied on a Naive Bayesian classifier yielding satisfactory performance for both polarity classification problems. The Naive Bayes classifier was also examined in Go et al. [5] in comparison to the Maximum Entropy and the Support Vector Machines (SVM). To represent tweets, the authors employed the term vector model together with a set of feature selection algorithms. They too examined both polarity classification problems.

Closer to our approach is the work of Pac et al. [10], where the authors employ the term vector model in conjunction with Part-of-Speech tagging. They consider the General Polarity Classification problem in the form of two phases: the first one distinguishes between subjective and objective tweets, while the second one distinguishes between negative and positive tweets. The applied their method on the SVM, the CRF and the Naive Bayes classification algorithms, with the last one achieving the highest performance. Davidov et al. [1] also employed patterns composed of frequent words and n-grams in their sentiment analysis approach. Instead of the typical polarity classes, though, they considered a diverse set of hashtag-based polarity classes. As a classification algorithm, they used a k-Nearest Neighbors-like strategy.

All these methods examine the problem of sentiment analysis with respect to a specific, predefined (set of) languages. None of them considers, therefore, Sentiment Analysis in language-independent settings. Most importantly, though, none of them has explored the use of n-gram graphs in this problem.

6. CONCLUSIONS

In this paper, we examined the use of the n-gram graphs model in the context of Sentiment Analysis over Social Media. Our thorough experimental study verified that it is the most suitable representation model for this task, due to a series of advantages it encompasses: first, it allows for fuzzy and substring matching - a functionality of primary importance in open domains like Twitter - and, second, it is a language-neutral method that makes no assumptions on the underlying languages. Apart from the resulting high effectiveness, though, this model exhibits high classification efficiency, as well. The reason is that it involves a limited number of features that solely depends on the corresponding number of classes. In the future, we plan to investigate its coupling with evidence drawn from the social graph of Twitter as well as with behavioral patterns of its users, thus providing a holistic, and highly effective solution to polarity classification.

7. ACKNOWLEDGEMENT

This work has been supported by the SocIoS project and has been partly funded by the European Commission's 7th Framework Programme through theme ICT-2009.1.2: Internet of Services, Software and Virtualisation under contract no.257774.

References

[1] D. Davidov, O. Tsur, and A. Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *COLING*, pages 241–249, 2010.

[2] J. Eisenstein, B. O'Connor, N. A. Smith, and E. P. Xing. A latent variable model for geographic lexical variation. In *EMNLP*, pages 1277–1287, 2010.

[3] G. Giannakopoulos, V. Karkaletsis, G. A. Vouros, and P. Stamatopoulos. Summarization system evaluation revisited: N-gram graphs. *TSLP*, 5(3), 2008.

[4] G. Giannakopoulos and T. Palpanas. Content and type as orthogonal modeling features: a study on user interest awareness in entity subscription services. *International Journal of Advances on Networks and Services*, 3(2), 2010.

[5] A. Go, L. Huang, and R. Bhayani. Twitter sentiment analysis. *Entropy*, 2009.

[6] S. D. Kamvar and J. Harris. We feel fine and searching the emotional web. In *WSDM*, pages 117–126, 2011.

[7] T. Lake and W. Fitzgerald. Twitter sentiment analysis. 2011.

[8] C. Manning, P. Raghavan, and H. Sch
"utze. *Introduction to information retrieval*, volume 1. Cambridge University Press, 2008.

[9] B. O'Connor, R. Balasubramanyan, B. R. Routledge, and N. A. Smith. From tweets to polls: Linking text sentiment to public opinion time series. In *ICWSM*, 2010.

[10] A. Pak and P. Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*, 2010.

[11] Y. Tausczik and J. Pennebaker. The psychological meaning of words: Liwc and computerized text analysis methods. *Journal of Language and Social Psychology*, 29(1):24, 2010.

[12] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *ICWSM*, 2010.

[13] T. Wilson, P. Hoffmann, S. Somasundaran, J. Kessler, J. Wiebe, Y. Choi, C. Cardie, E. Riloff, and S. Patwardhan. Opinionfinder: A system for subjectivity analysis. In *HLT/EMNLP*, 2005.

[14] I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann Pub, 2005.

[15] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, pages 177–186, 2011.